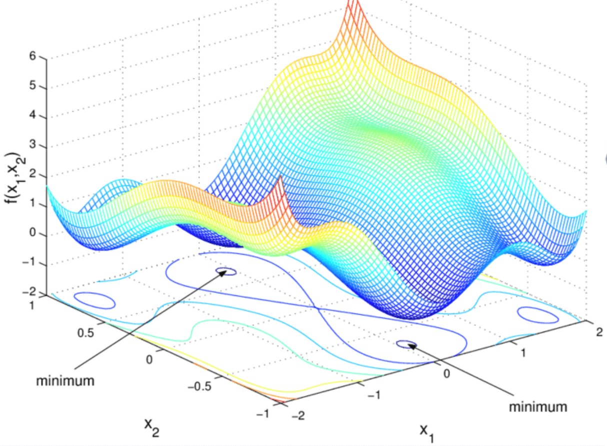
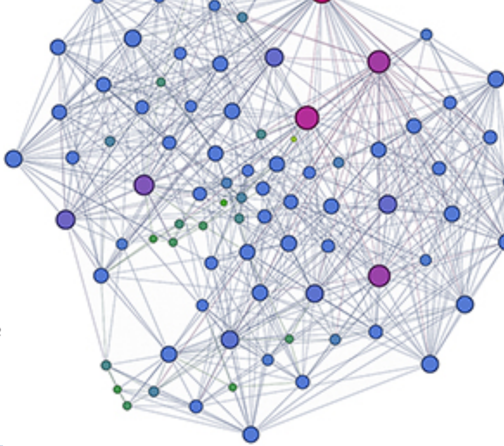




On Solving Large-Scale MINLPs

Hassan Hijazi

THE ARPA-E GRID OPTIMIZATION COMPETITION

Thanks to ARPA-E and PNNL (esp. Steve Elbert and Arun Veeramany)

Solving a Large-Scale Mixed-Integer Nonlinear Program

Only 97 pages to get the formulation right

Solving a Large-Scale Mixed-Integer Nonlinear Program

Only 97 pages to get the formulation right

At an abstract level, this is what we're dealing with:

$$
\begin{aligned}
&\min \ f(x, y) \\
&s.t. \\
&g(x, y) \leq 0, \\
&h(x, y) = 0, \\
&x \in \mathbb{R}^n, y \in \mathbb{Z}^m
\end{aligned}
$$

Solving a Large-Scale Mixed-Integer Nonlinear Program

Only 97 pages to get the formulation right

At an abstract level, this is what we're dealing with:

$$
\begin{aligned}
&\min \;\; f(x, y) \\
&s.t. \\
&g(x, y) \leq 0, \\
&h(x, y) = 0, \\
&x \in \mathbb{R}^n, y \in \mathbb{Z}^m
\end{aligned}
$$

Many many non-convex constraints!

Many many discrete and continuous variables!

How Many is Many Many?

Largest instance has 1,224,080,000 variables and 837,488,000 constraints.

How Many is Many Many?

Largest instance has 1,224,080,000 variables and 837,488,000 constraints.

This Being Said

Only a subset of these variables and constraints matter

Due to the flexibility of contingency constraints and the large cost of basecase objective

How Many is Many Many?

Largest instance has 1,224,080,000 variables and 837,488,000 constraints.

This Being Said

Only a subset of these variables and constraints matter

Due to the flexibility of contingency constraints and the large cost of basecase objective

Largest instance has 612,040 important variables and 418,744 important constraints.

How Many is Many Many?

Largest instance has 612,040 important variables and 418,744 important constraints.

Only a subset of these variables are free, the rest fall under "auxiliary" variables

Only a subset of these Constraints will be active, the rest fall under "redundant" constraints

How Many is Many Many?

Largest instance has 612,040 important variables and 418,744 important constraints.

Only a subset of these variables are free, the rest fall under “auxiliary” variables

Only a subset of these Constraints will be active, the rest fall under “redundant” constraints

Variable Projection

Lazy Constraint Generation

Down to ~50,000 variables and constraints

Discretely Dealing With Discreteness

Iterative Batch Rounding

Discretely Dealing With Discreteness

Iterative Batch Rounding

Inspired by MINLP heuristics such as Feasibility Pump[1] and Fix-and-Relax [2]

[1] M. Fischetti, F. Glover, and A. Lodi, "The feasibility pump," Mathematical Programming, vol. 104, no. 1, pp. 91–104, 2005

[2] G. Belvaux and L. A. Wolsey, "bc—prod: A specialized branch-and-cut system for lot-sizing problems," Management Science, vol. 46, no. 5, pp. 724–738, 2000

Discretely Dealing With Discreteness

Iterative Batch Rounding

Inspired by MINLP heuristics such as Feasibility Pump[1] and Fix-and-Relax [2]

Algorithm 1 Iterative Batch Rounding (IBR)

1: Group discrete variables into predefined batches $\mathcal{B}_1$ to $\mathcal{B}_n$.
2: Solve continuous relaxation of MINLP (1).
3: **for** $i \in \{1, \ldots, n\}$ **do**
4: Call the custom ROUND function on batch $\mathcal{B}_i$
5: Fix all rounded variables in batch $\mathcal{B}_i$
6: Solve the continuous relaxation of reduced MINLP (1).
7: **end for**

[1] M. Fischetti, F. Glover, and A. Lodi, "The feasibility pump," Mathematical Programming, vol. 104, no. 1, pp. 91–104, 2005

[2] G. Belvaux and L. A. Wolsey, "bc—prod: A specialized branch-and-cut system for lot-sizing problems," Management Science, vol. 46, no. 5, pp. 724–738, 2000

Discretely Dealing With Discreteness

Iterative Batch Rounding

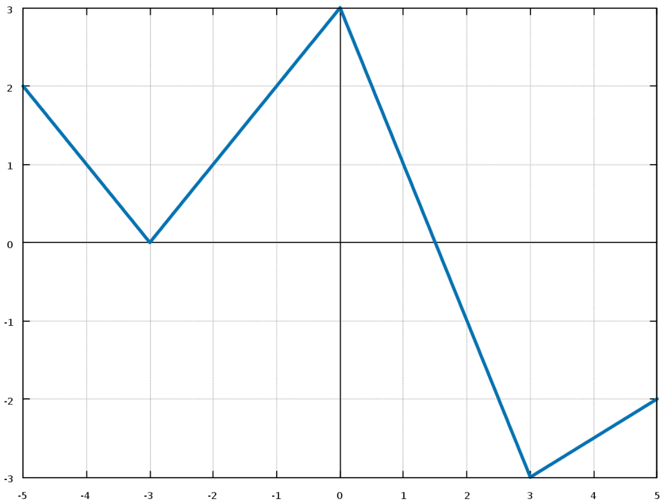
Algorithm 1 Iterative Batch Rounding (IBR)

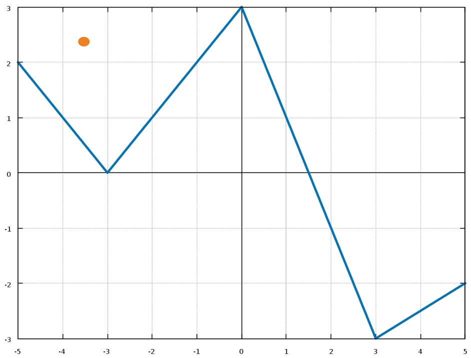
```
Group discrete variables into predefined batches B_1 to B_n.
Solve continuous relaxation of MINLP (1).
for i ∈ {1,...,n} do
    Call the custom ROUND function on batch B_i
    Fix all rounded variables in batch B_i
    Solve the continuous relaxation of reduced MINLP (1).
end for
```

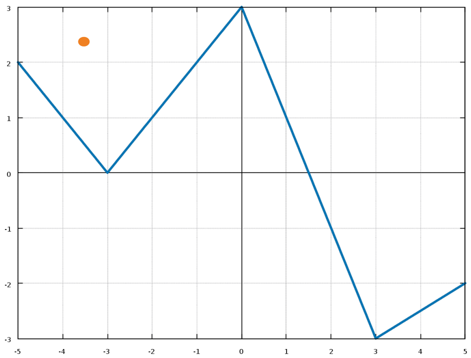
Two observations:

1. Different batch orderings can (dramatically) impact solution quality
2. Different rounding techniques can (dramatically) impact solution quality

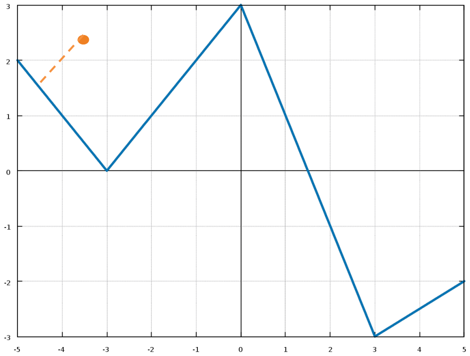
Rounding Binaries in Piecewise Linear Functions



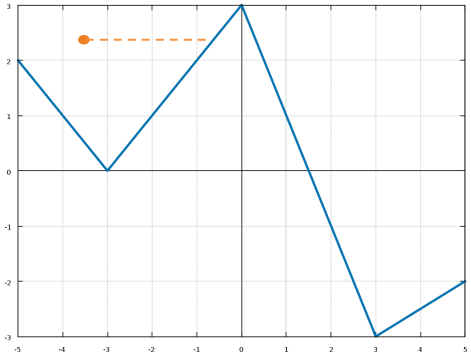

3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1
2
3
4
5



Orthogonal Projection



Fixed-Coordinate



Dealing With Numerical Convergence

One solver to rule them all:
IPOPT[3]

[1] A. Waechter and L. T. Biegler, “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,” Math. Programming, vol. 106, no. 1, pp. 25–57, 2006.

Dealing With Numerical Convergence

One solver to rule them all:
IPOPT[3]

IPOPT can be picky, preferring some formulations to others

[1] A. Waechter and L. T. Biegler, “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,” Math. Programming, vol. 106, no. 1, pp. 25–57, 2006.

Dealing With Numerical Convergence

One solver to rule them all:
IPOPT[3]

IPOPT can be picky, preferring some formulations to others

IPOPT can be moody, disliking some starting points and variable bounds

[1] A. Waechter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," Math. Programming, vol. 106, no. 1, pp. 25–57, 2006.

Dealing With Numerical Convergence

- Projection of auxiliary variables $p_g, \, p_j, \, q_j, \, \text{and} \; t_j$

  A good tradeoff between Jacobian/Hessian sparsity and number of equations

Dealing With Numerical Convergence

- Projection of auxiliary variables $p_g,\, p_j,\, q_j, \text{ and } t_j$

  A good tradeoff between Jacobian/Hessian sparsity and number of equations

- Dealing with non-differentiability $\sqrt{(p_e^o)^2 + (q_e^o)^2 + \epsilon} \leq \bar{\boldsymbol{r}}_e v_i + s_e + \epsilon$

Dealing With Numerical Convergence

- Projection of auxiliary variables p_g, p_j, q_j, and t_j

  A good tradeoff between Jacobian/Hessian sparsity and number of equations

- Dealing with non-differentiability $\sqrt{(p_e^o)^2 + (q_e^o)^2 + \epsilon} \leq \bar{\boldsymbol{r}}_e v_i + s_e + \epsilon$

- A good starting point and good bounds

$$\min\left(0, \underline{\boldsymbol{q}}_g\right) \leq q_g \leq \max\left(0, \bar{\boldsymbol{q}}_g\right) \tag{1}$$

$$-1.5\boldsymbol{r}_e \leq p_e^o \leq 1.5\boldsymbol{r}_e \tag{2}$$

$$-1.5\boldsymbol{r}_e \leq p_e^d \leq 1.5\boldsymbol{r}_e \tag{3}$$

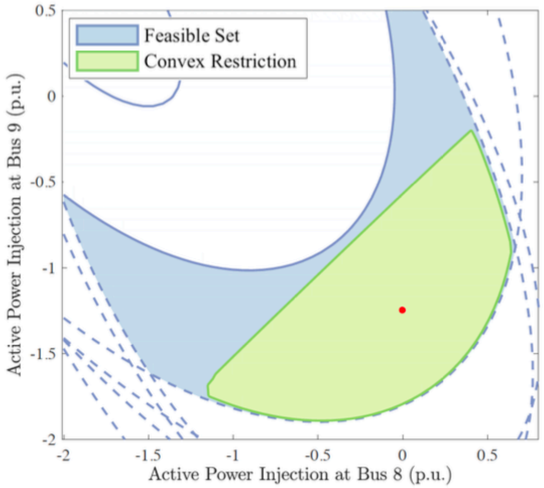
$$-1.5\boldsymbol{r}_f \leq p_f^o \leq 1.5\boldsymbol{r}_f \tag{4}$$

$$-1.5\boldsymbol{r}_f \leq p_f^d \leq 1.5\boldsymbol{r}_f \tag{5}$$

$$-2\pi \leq \theta_i \leq 2\pi \tag{6}$$

Work In Progress

- Projection of all auxiliary variables
- Building Convex Restrictions
- Building Convex Relaxations
- Lazy Constraint Generation for Contingencies

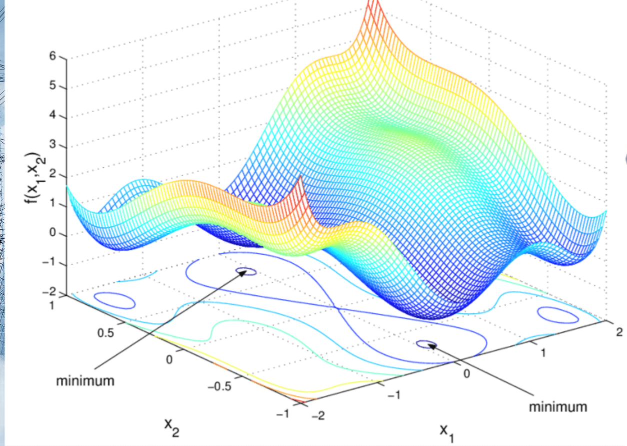


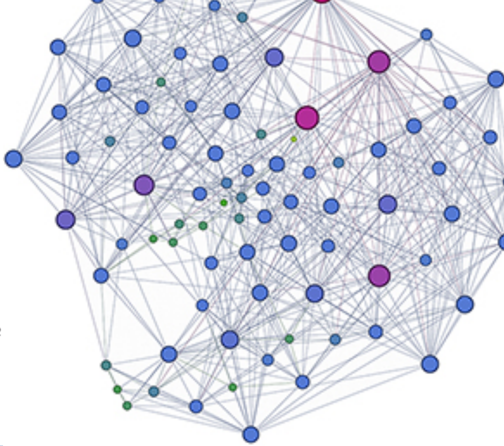
From “Convex Restriction of Power Flow Feasible Sets” by Lee et al.

Lee, Dongchan, et al. "Convex restriction of power flow feasibility sets." IEEE Transactions on Control of Network Systems 6.3 (2019): 1235-1245.

Things that Really Helped

- Starting modeling early-on (worked hard for Trial 1)
- Using a fast modeling language with symbolic differentiation and disjunctive constraint support (Gravity)
- Testing, testing and testing (a total of 2650 submissions to the sandbox)





Thanks!